Урок 2: Знакомство со средой разработчика

На предыдущем уроке мы узнали, как при помощи небольшого программного кода можно расширить функциональность Autodesk Revit.

На уроке 1 мы упоминали технологию .NET, которая позволяет приложениям взаимодействовать друг с другом. Если вы хотите узнать больше, то вам стоит прочитать раздел Дополнительные темы, который находится дальше по тексту.

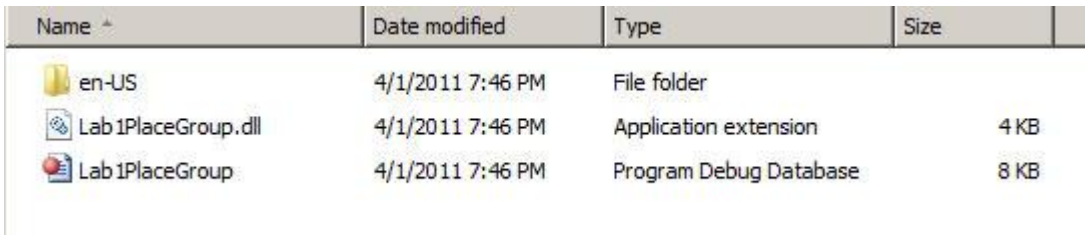
Давайте посмотрим поподробнее на то, что происходит при построении и выполнении программного кода из предыдущего занятия.

Обратная связь: напишите нам об этом уроке или обо всем курсе «Моя первая программа»: myfirstplugin@autodesk.com
(Пожалуйста, пишите на английском языке)

Что означает «построить» программный код?

Программный код, который вы набирали с помощью Visual C# Express на первом занятии, был написан в виде инструкций, понятных для человека (исходный код). Он должен быть переведен в код, который понятен для компьютера. При построении происходит следующее: на основе вашего кода строится файл DLL (Dynamic-Link Library), который может быть воспринят Autodesk Revit.

На следующей иллюстрации показана библиотека DLL с дополнительной базой для отладки (она содержит информацию, которая будет полезна при возникновении каких-либо ошибок в DLL). Эти файлы создаются во время построения исходного кода в Visual C# Express. Путь, по которому размещается скомпилированный файл DLL, указывается в настройках проекта Visual C# Express. По умолчанию компиляция осуществляется во вложенную папку проекта VisualC# Express.



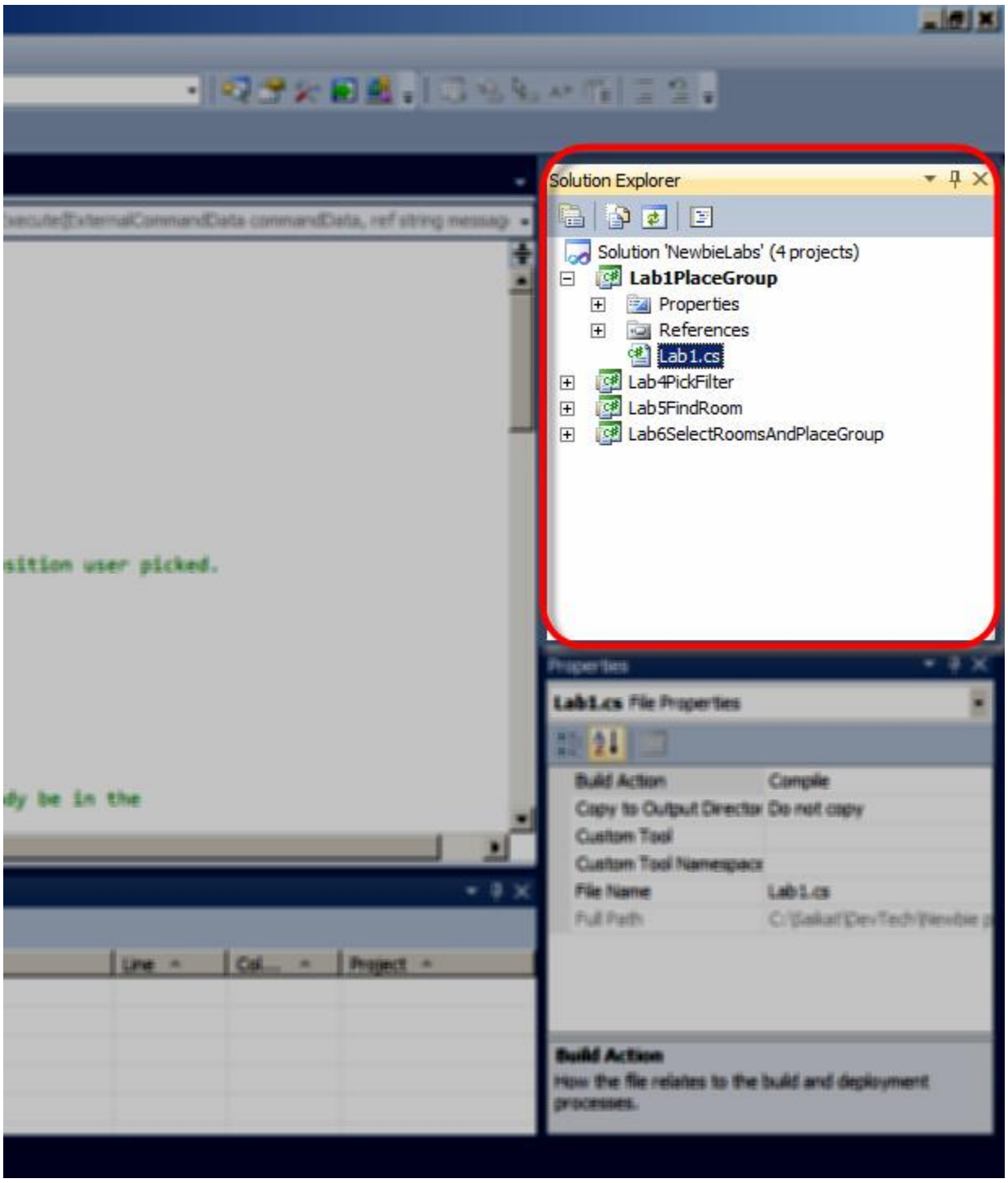
Выбор языка программирования и средств разработки

Как люди могут общаться на разных языках, так и разработку плагина для Revit можно вести на различных языках программирования. На данных занятиях мы используем C# – мощный язык программирования, наиболее популярный среди разработчиков Revit.

Существуют различные наборы инструментов для разработки кода на C# – от Sharp Develop с открытым исходным кодом до профессиональной среды разработки Visual Studio, являющейся флагманским продуктом Microsoft. В нашем случае мы будем использовать Visual C# Express – бесплатную версию Visual Studio.

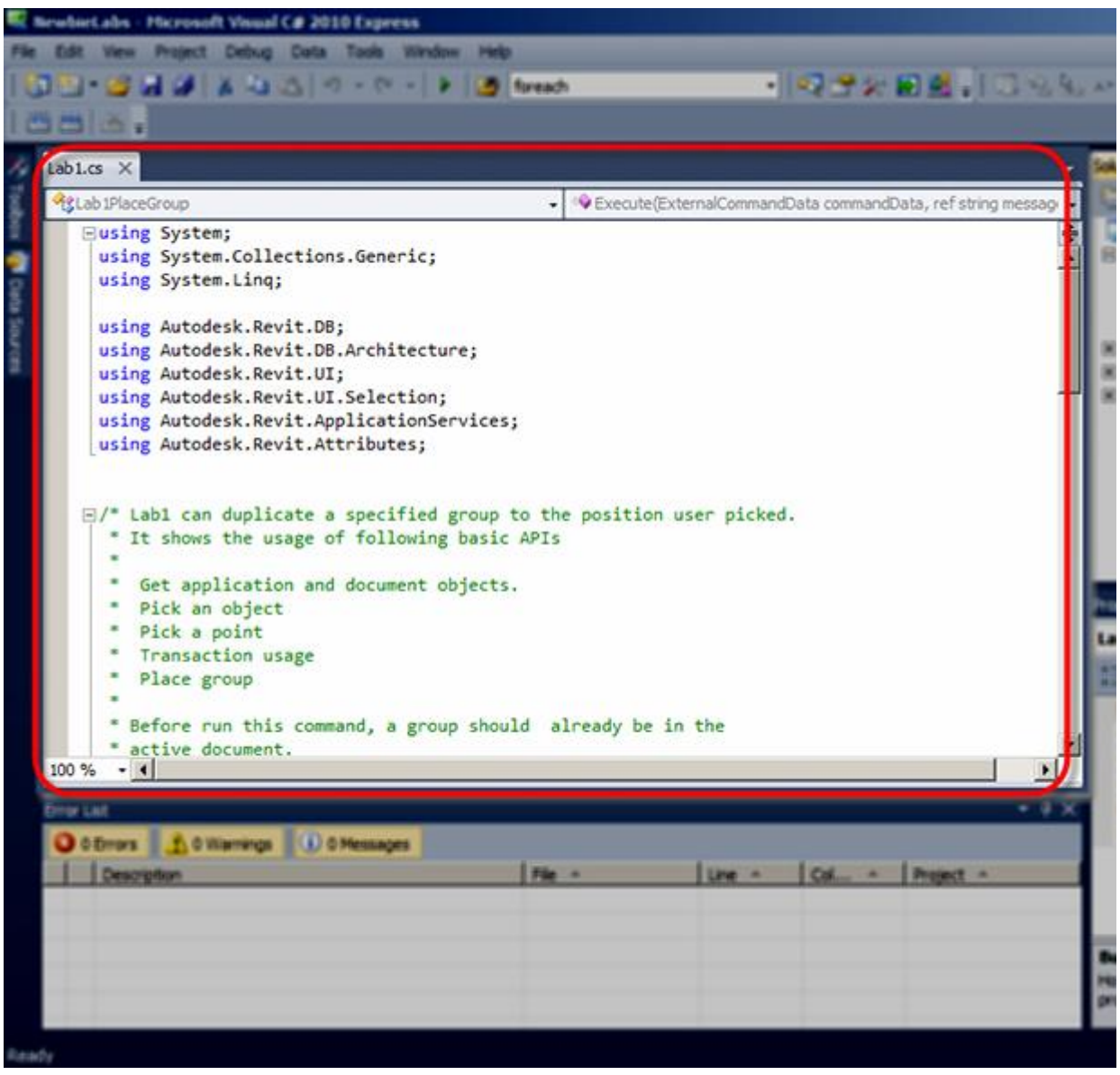
Visual C# Express – это встроенная среда разработки (IDE): она содержит инструменты, разделы, меню и панели инструментов для разработки программного кода.

Проект в среде Visual C# Express состоит из файлов Решения и файлов Проекта, а также из Элементов проекта – отдельных файлов, которые относятся к проекту. В решение могут входить один или несколько проектов. Каждый проект может содержать элементы проекта, такие как файлы ресурсов, значки и т.п. Большинство из них будут включены при построении в исполняемый файл (exe или dll). В Visual C# Express есть Обозреватель Решения (Solution Explorer), в котором все элементы решения организованы в виде древовидной структуры:

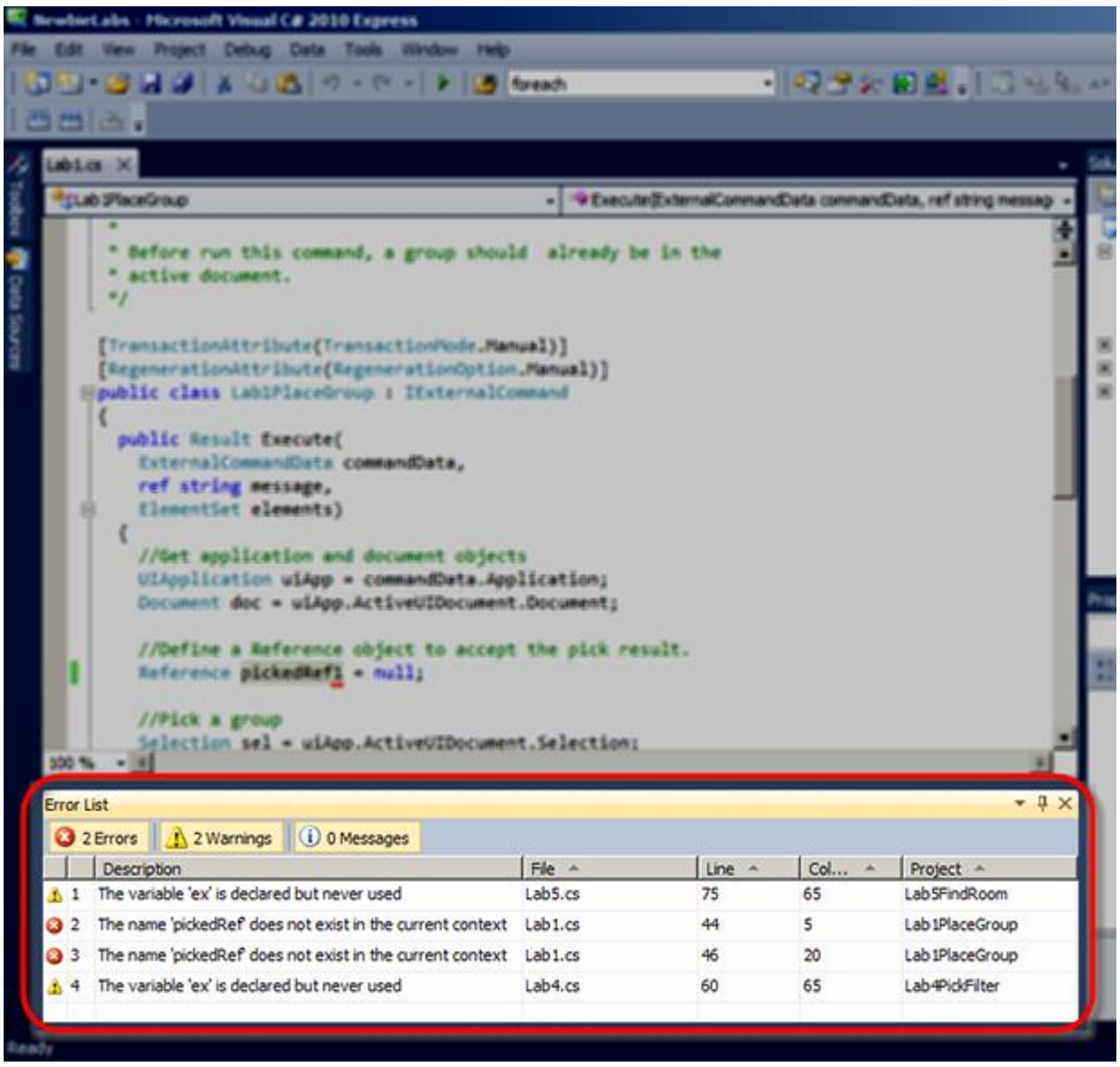


Среда Visual C# Express также содержит текстовый редактор и редактор пользовательского интерфейса. Они отображаются в главном окне в зависимости от того, какой файл в данный момент редактируется. В текстовом редакторе вводится программный код плагина для Revit на языке C#. Наряду со стандартными для текстового редактора особенностями (например: пометки и отображение номера строки), данный редактор предоставляет такие средства, как IntelliSense и сворачиваемые фрагменты кода.

IntelliSense, очень мощный инструмент в семействе редакторов Visual Studio, в разы ускоряет процесс программирования: он автоматически предлагает для выбора элементы кода, основываясь на уже введенных текстовых символах и доступных объектах.



Одной из основных особенностей Visual C# Express является возможность построения кода C# в исполняемый файл. Во время построения компилятор проводит ряд проверок и анализ кода: выявляет синтаксические ошибки языка C# в коде, проверяет, существует ли и определена ли соответствующая переменная, и т.п.. Обнаруженные ошибки выводятся в окно Списка ошибок, которое обычно находится в нижней части главного окна.

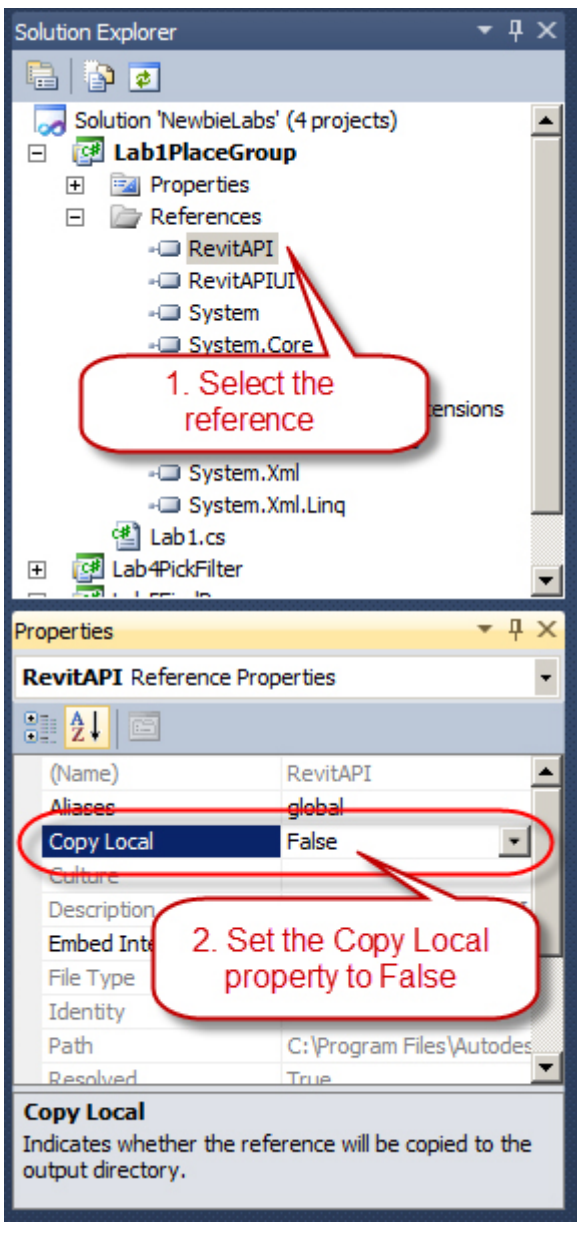


Обзор использования Visual C# Express

В этом разделе мы поэтапно проанализируем, как на предыдущем занятии велась работа с Visual C# Express. Мы рассмотрим процесс создания кода, опираясь на те знания о программировании, которые у вас уже есть.

1. Мы запустили Visual C# Express.
2. Затем мы создали проект библиотеки классов C#.

   Так как в этом учебном пособии для разработки используется язык C#, мы работаем в Visual C# Express. Именно поэтому мы видим **Visual C#** на вкладке **Установленные шаблоны (Installed Templates)** диалогового окна **Создать проект (New Project)**.



В средней части этого окна отображается несколько типов возможных для создания проектов. Мы выбрали шаблон проекта в соответствии с тем, какое приложение планировалось создать.

Плагины, загружаемые в Revit, должны представлять собой Библиотеки классов (DLL). Именно поэтому на втором этапе мы выбрали шаблон **Библиотеки классов (Class Library)**. Имя, присвоенное проекту, служит для обозначения проекта в рамках Решения.

3. Заготовка для проекта создаётся с уже загруженными базовыми ссылками на компоненты .NET, включая файл класса C#, код которого отображается в окне текстового редактора.

4. Сохранение Решения приводит к созданию на жестком диске файлов, содержащих в себе весь контент вашей библиотеки классов, что впоследствии позволит открыть эти файлы и продолжить редактирование.

5. Стандартный проект Visual C# Express не позволяет сразу использовать Revit API. Для этого нужно добавить ссылки на DLL-библиотеки интерфейса API Revit: *RevitAPI.dll* и *RevitAPIUI.dll*.

6. Для работы с Revit API чаще всего используются ссылки на две библиотеки DLL: одна предоставляет доступ к функциональности Revit, другая – к пользовательскому интерфейсу. Для работы с Revit API вы должны настроить эту связь.

- *RevitAPI.dll* содержит классы для доступа к данным приложения: документам, параметрам и т.п.
- *RevitAPIUI.dll* содержит классы для взаимодействия с пользовательским интерфейсом, включая команды, инструменты выбора и диалоговые окна.

После добавления ссылок на эти объекты лучше сразу же настроить их свойства.

По умолчанию Visual C# Express назначает всем ссылкам параметра **Копировать локально (CopyLocal)** значение **Да (True)**. Это означает, что ссылочные файлы будут скопированы в выходную папку при построении. В нашем случае требуется изменить

значение этого параметра на **Нет (False)**, чтобы данные библиотеки не копировались вместе с вашей сборкой.

Для того чтобы изменить значение этого параметра для DLL, выберите Revit API DLL в разделе **Ссылки (References)** окна **Обозревателя решений (Solution Explorer)** с правой стороны от окна Visual Studio. В расположенном ниже окне **Свойства (Properties)** отображаются параметры ссылки на DLL. Среди них присутствует параметр **Копировать локально (Copy Local)**. Для изменения его значение с True на False раскройте выпадающий список и выберите соответствующее значение. Повторите те же действия для другой DLL.

Это нужно сделать по двум причинам. Во-первых, локальные копии занимают место на жестком диске и требуют времени для копирования; но, что более важно, Общеязыковой среде исполнения (CLR) может стать не понятно, какой экземпляр каждого из DLL-файлов должен быть загружен, если они существуют во многих местах.
Если библиотеки DLL не копируются локально в проект, CLR берет их из папки Revit, а это – именно то, что требуется.

7. На следующем этапе в проект добавляется программный код C#, использующий API Revit. Другими словами, Revit получает от нас инструкции по функционалу копирования выбранных пользователем групп из одного места в другое.

Хорошая идея во время доработки кода – **Построение Решения(Build Solution)** время от времени для того, чтобы проверить, не появились ли в программном коде ошибки. Когда строится Решение, код не должен обязательно быть законченным или функциональным. Такой подход может помочь избежать потенциальных проблем, всплывающих тогда, когда написание кода уже завершено, и требующих много времени на исправление. Отметим еще один полезный побочный эффект: перед построением Решения всегда выполняется автоматическое сохранение файлов с исходным кодом на диск.

Для того, чтобы построить Решение в Visual C# Express, выберите **Построить решение (Build Solution)** из всплывающего меню **Отладка** (**Debug).**

Примечание: вы можете использовать функциональную клавишу F6, чтобы быстро построить решение, не заходя в меню.

Если процесс сборки прошел успешно, в строке состояния Visual C# Express появляется сообщение **Построение успешно завершено (Build Succeeded)**.

Итак, на этом занятии вы увидели, что происходит, когда строится проект, а также получили базовую информацию о C# и Visual C# Express. Были более детально проанализированы этапы построения базового плагина для Revit, созданного нами на занятии 1.

Дополнительные темы

Что такое .NET?

.NET Framework – это набор программных модулей, который устанавливается в операционной системе Microsoft® Windows®* и предоставляет основную платформу, библиотеки и службы для всех .NET приложений. Службы в основном включают управление памятью, сборку мусора, систему общих типов, библиотеку классов и т.д.

**Подмножества .NET также доступны в других операционных системах либо через открытый код Mono project, либо через Microsoft® Silverlight®, но это выходит за рамки данного учебного пособия. Мы сосредоточимся только на использовании .NET в контексте Microsoft Windows.*

Что включает в себя .NET Framework?

Framework включает в себя два основных компонента:

1. *Common Language Runtime (CLR)* – агент (или выполняющий движок) в .NET Framework, предназначенный для управления исполняемым кодом. Код, который написан и выполняется в этом цикле, также называется управляемым кодом. Все управляемые коды запускаются под контролем CLR. CLR управляет кодом путем предоставления основных служб, таких как управление памятью (в том числе освобождение ранее занятой памяти, когда она больше не нужна), обработка ошибок (или исключений), управление использованием многоканального потока исключений и обеспечение соблюдения правил для различных типов объектов. CLR, в действительности, – это основа .NET Framework.

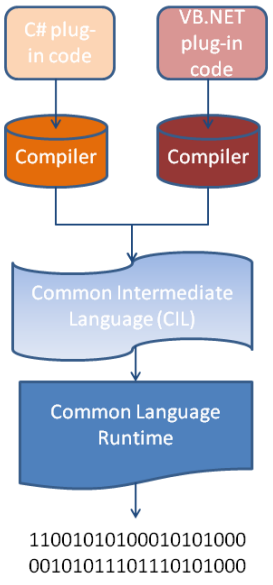
2. Библиотека классов *.NET Framework Class Library* – как понятно из названия, это библиотека или набор типов объектов, который может использоваться из вашего собственного кода, когда разрабатываются приложения .NET. Эти приложения .NET предназначены для работы в Windows (с интерфейсом командной строки или с графическим пользовательским интерфейсом), в Интернете и на мобильных устройствах. Эта библиотека доступна для всех языков, использующих .NET Framework.

Как упоминалось выше, CLR улучшает устойчивость кода, проверяя, что выполняемый код согласуется с common type system (CTS). CTS обеспечивает, что весь .NET (или управляемый) код, независимо от языка, использует одинаковый набор типов объектов и может работать вместе в одной и той же среде. Именно эта особенность делает возможным написание приложений на языке разработки на ваш выбор, а также позволяет применять компоненты и код, написанные на других .NET-совместимых языках.

Построение исполняемых файлов

Когда вы строите код в DLL, он компилируется в *Common Intermediate Language* (CIL – также известен как MSIL), используя ориентированный на конкретный язык компилятор. CIL – это не зависящий от процессора набор инструкций, которые могут быть выполнены с помощью CLR в операционной системе Windows. CIL, как правило, может переноситься на 32- и 64-битные платформы и, даже, в некоторой степени, на операционные системы, отличные от Windows. Код CIL, сгенерированный из вашего исходного кода C#, далее пакуется в сборку .NET. Такая сборка – это библиотека кода CIL, хранимая в формате *Portable Executable* (PE) (которая содержит одновременно и CIL, и связанные с ним метаданные). Сборки могут быть как исполняемыми сборками (EXE), так и библиотечными сборками (DLL).

В этом учебном пособии плагины Revit компилируются в файлы библиотечных сборок (DLL), которые далее загружаются и выполняются внутри пространства памяти Revit.



Запуск исполняемых файлов

Во время выполнения сборки .NET, находящийся в этой сборке код CIL проходит через JIT-компилятор CLR, для того, чтобы сгенерировать машинный код. Компиляция JIT из CIL в машинный код происходит, когда приложение выполняется. Если не весь код требуется во время выполнения, компилятор JIT конвертирует CIL только тогда, когда он нужен, тем самым сохраняя время и память. Он также сохраняет и генерирует код в памяти, делая доступным последующее его использование без перекомпиляции.

На последнем этапе процесса машинный код исполняется процессором компьютера.

Если вы хотите получить более детальную информацию по процессу построения приложений .NET, обратитесь к MSDN Library.